Virtual Disk Generation Linking

Stuart Maclean
Applied Physics Laboratory
University of Washington

October 2015

1 Introduction

See also the VMWare 'vmdk_specs.pdf' file, which contains a field-by-field description of the VMWare 'Descriptor File' format.

2 Virtual Box VDI Files

3 VMWare VMDK Files

VMDKFile metadata produced by vmvols tool 'vmdkinfo'. Prints SparseExtentHeader and Descriptor info at head of any .vmdk file.

We have examples of various standalone (single generation) and parent-child relationships across .vmdk files where

- both initial disk and snapshot created within a VMWare host-based product, e.g. Workstation.
- both initial disk and snapshot created within VirtualBox, using the .vmdk hard drive file type (and not the native VDI format)
- the initial disk is via an VirtualBox import operation of an ovf package.
- we also examine the streamOptimized vmdk file variant used in ovf/ova packages (an output of e.g. packer)

3.1 Disks created by VMWare Workstation

First generation of a virtual machine hard drive. VM generated by VMWare Workstation Pro 12 (trial edition), Oct 2015.

```
$ vmdkinfo ~/vmware/Windows\ 7\ x64/Windows\ 7\ x64.vmdk

Flags: 00000003
```

```
Version: 1
Capacity: 125829120
GrainSize: 128
DescriptorOffset: 1
DescriptorSize: 20
NumGTEsPerGT: 512
rgdOffset: 21
gdOffset: 7716
Overhead: 15488
Compression: 0

# Disk DescriptorFile
version=1
encoding="UTF-8"
CID=7c77be3e
parentCID=ffffffff
isNativeSnapshot="no"
createType="monolithicSparse"

# Extent description
RW 125829120 SPARSE "Windows 7 x64.vmdk"

# The Disk Data Base
#DDB

ddb.adapterType = "lsilogic"
ddb.geometry.cylinders = "7832"
ddb.geometry.heads = "255"
ddb.geometry.sectors = "63"
ddb.longContentID = "998e1e01d9a37fe7ae03f9e17c77be3e"
ddb.uuid = "60 00 C2 92 c7 b3 76 ce-43 c5 92 ee 09 64 6c 58"
ddb.virtualHWVersion = "12"
```

3.2 Snapshot Disks created by VMWare Workstation

Snapshot (second generation) of the virtual machine hard drive above. VM generated by VMWare Workstation Pro 12 (trial edition), Oct 2015. Note how little/vague the 'parent pointer' is, just a 'hint' in the descriptor.

```
$ vmdkinfo ~/vmware/Windows\ 7\ x64/Windows\ 7\ x64-000001.vmdk

Flags: 00000003
Version: 1
Capacity: 125829120
GrainSize: 128
DescriptorOffset: 1
```

```
DescriptorSize: 20
NumGTEsPerGT: 512
rgdOffset: 21
gdOffset: 7716
Overhead: 15488
Compression: 0

# Disk DescriptorFile
version=1
encoding="UTF-8"
CID=7c77be3e
parentCID=7c77be3e
isNativeSnapshot="no"
createType="monolithicSparse"
parentFileNameHint="/home/stuart/vmware/Windows 7 x64/Windows 7 x64.vmdk"
# Extent description
RW 125829120 SPARSE "Windows 7 x64-000001.vmdk"

# The Disk Data Base
#DDB
```

Heuristic: If descriptor has parentFileNameHint=entry, use it in locating a parent. The CID and parentCID in the child are equal, and both are equal to the CID in the parent file. Use this fact as an extra check?

3.3 Further Snapshot Disks created by VMWare Workstation

Additional snapshot (third generation) of the virtual machine hard drive above in examples 1 and 2. VM generated by VMWare Workstation Pro 12 (trial edition), Oct 2015. Note how little/vague the 'parent pointer' is, just a 'hint' in the descriptor.

```
[~/.../infosec/vmvols/cli (master)]$ ./vmdkinfo ~/vmware/Windows\ 7\ x64/Windows\ 7\ x64-000
Flags: 00000003
Version: 1
Capacity: 125829120
GrainSize: 128
DescriptorOffset: 1
DescriptorSize: 20
NumGTEsPerGT: 512
rgdOffset: 21
gdOffset: 7716
Overhead: 15488
Compression: 0
```

```
# Disk DescriptorFile
version=1
encoding="UTF-8"
CID=7c77be3e
parentCID=7c77be3e
isNativeSnapshot="no"
createType="monolithicSparse"
parentFileNameHint="/home/stuart/vmware/Windows 7 x64/Windows 7 x64-000001.vmdk"
# Extent description
RW 125829120 SPARSE "Windows 7 x64-000002.vmdk"

# The Disk Data Base
#DDB
```

Note the parentFileNameHint in example 3, as was in example 2. Note also that example 3 appears to have no 'DDB' entries, same as example 2.

3.4 Disks of a Cloned VM in VMware Workstation

Clone of the VM containing the disk mentioned above in examples 1-3. Clone operation done in same VMware Workstation product as used to build the 1-3 examples disk. Note the somewhat odd .vmdk name, containing 'cl' for 'clone'?

```
[~/.../infosec/vmvols/model (master)]$ ../cli/vmdkinfo ~/vmware/Clone\ of\ Windows\ 7\ x64/W
Flags: 00000003
Version: 1
Capacity: 125829120
GrainSize: 128
DescriptorOffset: 1
DescriptorSize: 20
NumGTEsPerGT: 512
rgdOffset: 21
gdOffset: 7716
Overhead: 15488
Compression: 0

# Disk DescriptorFile
version=1
encoding="UTF-8"
CID=7c77be3e
parentCID=ffffffff
isNativeSnapshot="no"
createType="monolithicSparse"

# Extent description
RW 125829120 SPARSE "Windows 7 x64-cl1.vmdk"
```

```
# The Disk Data Base
#DDB

ddb.adapterType = "lsilogic"
ddb.deletable = "true"
ddb.geometry.cylinders = "7832"
ddb.geometry.heads = "255"
ddb.geometry.sectors = "63"
ddb.longContentID = "ba254740a7f06f67777b34661a687c32"
ddb.uuid = "60 00 C2 91 ac 9e 55 71-5a a4 b4 1c 71 48 8f be"
ddb.virtualHWVersion = "12"
```

Note how the CID is actually the same as in the VM from which this VM was cloned!! This is hopeless! The CID cannot be used as a definitive identifier for a VMDK disk! Hint: Use the db.uuid instead?? Or some combo of the two??

3.5 Disks created by Packer

Standalone vmdk produced by packer (v0.8.6) with a 'virtualbox-iso' builder. Note that even though the configuration is for virtualBox, the file format for the virtual disk is a .vmdk. Note how the createType is 'streamOptimized'. The .vmdk file is associated with a .ovf file (found alongside the .vmdk). Note also how packer adds the 'ddb.uimage' fields into the descriptor. These will be used in VirtualBox imports at least.

```
$ ./vmdkinfo ~/apl/projects/infosec/packer-vms/ubuntu-12.04.4-amd64/base/products/ubuntu-12.

Flags: 00030001
Version: 3
Capacity: 81920000
GrainSize: 128
DescriptorOffset: 1
DescriptorSize: 2
NumGTEsPerGT: 512
rgdOffset: 0
gdOffset: -1
Overhead: 128
Compression: 1

# Disk DescriptorFile
version=1
CID=9f5528be
parentCID=ffffffff
createType="streamOptimized"
```

```
# Extent description
RDONLY 81920000 SPARSE "ubuntu-12.04.5-amd64-base-disk1.vmdk"

# The disk Data Base
#DDB

ddb.virtualHWVersion = "4"
ddb.adapterType="ide"
ddb.geometry.cylinders="16383"
ddb.geometry.heads="16"
ddb.geometry.sectors="63"
ddb.geometry.biosCylinders="1024"
ddb.geometry.biosHeads="255"
ddb.geometry.biosSectors="63"
ddb.uuid.image="ba1d7b83-2e83-4777-90fb-61c8251ccd69"
ddb.uuid.parent="00000000-0000-0000-0000-000000000000"
ddb.uuid.modification="00000000-0000-0000-0000-000000000000"
ddb.uuid.parentmodification="00000000-0000-0000-0000-000000000000"
ddb.comment=""
```

Note the ddb.uuid.* entries in the DDB section. Not sure what these are for?

Recall that this streamOptimized form of VMDK file is suitable for OVF/OVA packages but not for direct attachment into a VM. Instead, its OVF/OVA package must be first imported by the VM engine. The import process reads the streamOptimized input, but writes e.g. a monolithicSparse version locally as the VM's actual virtual disk format.

3.6 VMDK Disks created by/in VirtualBox

Created in/by VirtualBox, but by selecting 'VMDK' as the 'Hard Drive Format Type' in the VM creation wizard.

```
$ ./vmdkinfo  ~/VirtualBox\ VMs/Blank_VMDK/Blank_VMDK.vmdk

Flags: 00000003
Version: 1
Capacity: 268435456
GrainSize: 128
DescriptorOffset: 1
DescriptorSize: 20
NumGTEsPerGT: 512
rgdOffset: 21
gdOffset: 16437
Overhead: 32896
```

```
Compression: 0

# Disk DescriptorFile
version=1
CID=fe21c26a
parentCID=ffffffff
createType="monolithicSparse"

# Extent description
RW 268435456 SPARSE "Blank_VMDK.vmdk"

# The disk Data Base
#DDB

ddb.virtualHWVersion = "4"
ddb.adapterType="ide"
ddb.uuid.image="c86e611c-1092-48b0-b257-3e9480018efa"
ddb.uuid.parent="00000000-0000-0000-0000-000000000000"
ddb.uuid.modification="00000000-0000-0000-0000-000000000000"
ddb.uuid.parentmodification="00000000-0000-0000-0000-000000000000"
```

3.7 Snapshots in VirtualBox of VMDK Disks created in VirtualBox

A snapshot, taken by/in VirtualBox, of the .vmdk file described above (previous subsection):

```
$ ./vmdkinfo  ~/VirtualBox\ VMs/Blank_VMDK/Snapshots/\{b23c64d7-e938-4f7b-bc80-c56f4390dfe7\

Flags: 00000003
Version: 1
Capacity: 268435456
GrainSize: 128
DescriptorOffset: 1
DescriptorSize: 20
NumGTEsPerGT: 512
rgdOffset: 21
gdOffset: 16437
Overhead: 32896
Compression: 0

# Disk DescriptorFile
version=1
CID=f5050853
parentCID=ffffffff
createType="monolithicSparse"
```

```
# Extent description
RW 268435456 SPARSE "{b23c64d7-e938-4f7b-bc80-c56f4390dfe7}.vmdk"

# The disk Data Base
#DDB

ddb.virtualHWVersion = "4"
ddb.adapterType="ide"
ddb.uuid.image="b23c64d7-e938-4f7b-bc80-c56f4390dfe7"
ddb.uuid.parent="c86e611c-1092-48b0-b257-3e9480018efa"
ddb.uuid.modification="00000000-0000-0000-0000-000000000000"
ddb.uuid.parentmodification="00000000-0000-0000-0000-000000000000"
```

Note how the CID and parentCID in this descriptor do *not* given any hint on how to locate/identify the parent. The CIDs of the example disks 5 and 5 appear unrelated.

What *is* useful are the ddb.uuid.* entries, which VirtualBox seems to use whereas VMware products do not. The parent-child relationship between example disk 5 and 6 can be seen in the uuid.parent of disk 6 and uuid.image of disk 5. Recall that in VirtualBox's native VDI format, uuidImage and uuidParent are both fields present in the 'VDI header', so it's almost as if VirtualBox are shoe-horning their own parent/child linking info idea into a VMDK Descriptor file.

Heuristic: If descriptor has ddb.uuid.parent entry, use it in locating a parent. Such a parent will have a ddb.uuid.image descriptor entry whose value matches the uuid.parent entry in the child.

3.8 Disk created by VirtualBox via Import of OVF/OVA

VirtualBox import of the OVF package created for Example Disk 5:

```
$ ./vmdkinfo ~/VirtualBox\ VMs/ubuntu-12.04.5-amd64-base/ubuntu-12.04.5-amd64-base-disk1.vmd

Flags: 00000003
Version: 1
Capacity: 81920000
GrainSize: 128
DescriptorOffset: 1
DescriptorSize: 20
NumGTEsPerGT: 512
rgdOffset: 21
gdOffset: 5031
Overhead: 10112
Compression: 0
```

```
# Disk DescriptorFile
version=1
CID=6704c82b
parentCID=ffffffff
createType="monolithicSparse"

# Extent description
RW 81920000 SPARSE "ubuntu-12.04.5-amd64-base-disk1.vmdk"

# The disk Data Base
#DDB

ddb.virtualHWVersion = "4"
ddb.adapterType="ide"
ddb.geometry.cylinders="16383"
ddb.geometry.heads="16"
ddb.geometry.sectors="63"
ddb.geometry.biosCylinders="1024"
ddb.geometry.biosHeads="255"
ddb.geometry.biosSectors="63"
ddb.uuid.image="429e7834-80be-4bf5-a72f-a16c18cde00d"
ddb.uuid.parent="00000000-0000-0000-0000-000000000000"
ddb.uuid.modification="00000000-0000-0000-0000-000000000000"
ddb.uuid.parentmodification="00000000-0000-0000-0000-000000000000"
ddb.comment=""
```